Optimierungsprobleme

Instanz eines Optimierungsproblems

- zulässiger Bereich (meist implizit definiert)
- Zielfunktion
- Optimierungsrichtung $\text{opt} \in \{\max, \min\}$

Optimierungsproblem

- Menge von Instanzen
- meist implizit definiert

Beispiel (Kürzeste Wege)

Input:
- Digraph $G = (V, E)$
- Kantengewichte $\ell : E \to \mathbb{R}_{\geq 0}$
- zwei Knoten $s, t \in V$
- Definition „Weg“

Aufgabe: Finde einen bzgl. ℓ kürzesten s-t-Weg in G oder stelle fest, dass es keinen solchen Weg gibt.

Ausgabe:
- ein kürzester s-t-Weg
- Länge eines kürzesten s-t-Wegs
- „Es gibt keinen s-t-Weg in G.“

Optimierungs- und Entscheidungsprobleme

optimale Lösung	➡	function problem
optimaler Zielfunktionswert	➡	evaluation problem
Lösung mit Mindestwert L?	➡	recognition problem
zulässige Lösung?	➡	feasibility problem

$\left.\begin{array}{l}\text{recognition problem}\\ \text{feasibility problem}\end{array}\right\}$ Ja/Nein

Antwort „Ja“ oder „Nein“ ➡ Entscheidungsproblem

Algorithmen

Algorithmus

- gewünschte Ausgabe (Lösung)
- endliche Laufzeit, endliche Ressourcen

Laufzeit

- Anzahl elementarer Operationen
- Funktion der Codierungslänge
- abhängig von Instanz
- meist worst case-Abschätzung

Codierungslänge

- Darstellung im Rechner ➡ Binärcodierung
- Codierungslänge $=$ benötigte Bitanzahl

Beispiel (ganze Zahl)

- Zahl $\eta \in \mathbb{Z}$
- Vorzeichen $+$ Binärdarstellung
- Codierungslänge $\text{size}(\eta) = 1 + \lceil \log(|\eta| + 1) \rceil$
- Größenordnung $\mathcal{O}(\log(|\eta|))$

Codierungslänge

- Darstellung im Rechner ➡ Binärcodierung
- Codierungslänge ➡ benötigte Bitanzahl

Beispiel (LP)

- LP $\max c^T x$, $Ax \leq b$ mit $c \in \mathbb{Z}^n$, $A \in \mathbb{Z}^{m \times n}$
- Richtung + Zahl n + Zahl m + $(n + m + mn)$ Zahlen
- Codierungslänge
  $1 + \text{size}(n) + \text{size}(m) + (n + m + mn)\,\text{size}\left(\max\left\{|c_i|, |b_j|, |a_{ij}|\right\}\right)$
- Größenordnung $\mathcal{O}(mn)$

Polynomiell vs. exponentiell

Was ist eine gute Laufzeit? (Operation = Nanosekunde)

n	$100n\log(n)$	$10n^2$	$n^{\log(n)}$	2^n
10	3 µs	1 µs	2 µs	1 µs
20	9 µs	4 µs	420 µs	1 ms
50	28 µs	25 µs	4 s	13 d
100	66 µs	100 µs	5 h	$4 \cdot 10^{13}$ a
500	448 µs	2.5 ms	500 000 a	$1 \cdot 10^{134}$ a
1000	1 ms	10 ms	$3 \cdot 10^{13}$ a	$3 \cdot 10^{284}$ a
10^6	2 s	3 h	$1 \cdot 10^{103}$ a	$3 \cdot 10^{301\,013}$ a

Erdalter: $4,6 \cdot 10^9$ a Alter Universum: $13,7 \cdot 10^9$ a

Atome im Universum: 10^{78}

Die Klasse $\mathcal{P}$

Definition (Polynomieller Algorithmus)

- Entscheidungsproblem Π mit Algorithmus $\mathcal{A}$
- $\mathcal{A}$ heißt polynomiell, wenn es ein Polynom p gibt, so dass für jede Instanz I gilt:

$$\text{Laufzeit}\big(\mathcal{A}(I)\big) \leq p\big(\text{size}(I)\big)$$

- Polynomialität ist transitiv!

Definition (Klasse $\mathcal{P}$)

$$\mathcal{P} = \{\text{Problem } \Pi : \exists \text{ polynomieller Algorithmus für } \Pi\}$$

Polynomielle Reduktionen

Definition (Orakel)

- Entscheidungsproblem Π
- Funktion ω heißt Orakel für Π, wenn:
  - I Instanz von Π ➡ $\omega(I)$ Lösung
  - es gibt Polynom p, so dass für alle Instanzen I gilt: $\text{size}\big(\omega(I)\big) \leq p\big(\text{size}(I)\big)$

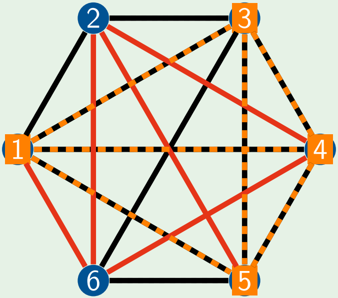
Definition (Polynomielle Reduktion)

- Entscheidungsprobleme Π_1 und Π_2, Orakel ω_2 für Π_2
- Π_1 heißt polynomiell reduzierbar auf Π_2 ($\Pi_1 \prec_p \Pi_2$), wenn Algorithmus für Π_1 existiert mit:
  - polynomiell viele elementare Operationen
  - vorkommende Zahlen polynomiell beschränkt
  - polynomiell viele Aufrufe von ω_2

Beispiel (Reduktion CLIQUE auf STABLE SET)

Input: Graph $G = (V, E)$, $K \in \mathbb{N}$

Frage: Gibt es Clique mit $\geq K$ Knoten?



Polynomielle Äquivalenz

Satz

„Π_1 ist polynomiell reduzierbar auf Π_2 ($\Pi_1 \prec_p \Pi_2$)“ bedeutet:

- $\Pi_2 \in \mathcal{P}$ ➡ $\Pi_1 \in \mathcal{P}$
- Π_1 *ist nicht schwerer als* Π_2

Ist außerdem $\Pi_2 \prec_p \Pi_1$

➡ Π_1 *und* Π_2 *sind „gleich schwer“*

Definition

Entscheidungsprobleme Π_1 und Π_2 heißen polynomiell äquivalent, wenn $\Pi_1 \prec_p \Pi_2$ und $\Pi_2 \prec_p \Pi_1$.

Zertifikate

Definition

- I Instanz eines Entscheidungsproblems
- Z Sequenz rationaler Zahlen
- size(Z) polynomiell beschränkt in size(I)

➡ Z heißt Zertifikat für I

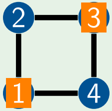
- Sinn: Zertifikat „beweist“ Richtigkeit der Antwort (Ja/Nein)
- häufig: polynomielle Codierung einer Lösung

Beispiel (STABLE SET)

Input: Graph $G = (V, E)$, $K \in \mathbb{N}$

Frage: Gibt es stabile Menge mit $\geq K$ Knoten?

Zertifikat: stabile Menge S mit $|S| \geq K$



Die Klasse $\mathcal{NP}$

Definition ($\mathcal{NP}$-Problem)

Entscheidungsproblem Π heißt $\mathcal{NP}$-Problem, wenn es einen Algorithmus $\mathcal{A}$ mit folgenden Eigenschaften gibt:

- Input für $\mathcal{A}$: Instanz I von Π und Zertifikat Z für I
- Laufzeit von $\mathcal{A}$ polynomiell in size(I) für jede Instanz I
- I NEIN-Instanz ➡ $\mathcal{A}(I, Z) =$ „NEIN“ für jedes Zertifikat
- I JA-Instanz ➡ $\exists$ Zertifikat Z mit $\mathcal{A}(I, Z) =$ „JA“

Definition (Klasse $\mathcal{NP}$)

$$\mathcal{NP} := \{\Pi : \Pi \text{ ist } \mathcal{NP}\text{-Problem}\}$$

Die Klasse co-$\mathcal{NP}$

Definition (co-$\mathcal{NP}$-Problem)

Entscheidungsproblem Π heißt co-$\mathcal{NP}$-Problem, wenn es einen Algorithmus $\mathcal{A}$ mit folgenden Eigenschaften gibt:

- Input für $\mathcal{A}$: Instanz I von Π und Zertifikat Z für I
- Laufzeit von $\mathcal{A}$ polynomiell in size(I) für jede Instanz I
- I JA-Instanz ➡ $\mathcal{A}(I, Z) =$ „JA“ für jedes Zertifikat
- I NEIN-Instanz ➡ $\exists$ Zertifikat Z mit $\mathcal{A}(I, Z) =$ „NEIN“

Definition (Klasse co-$\mathcal{NP}$)

$$\text{co-}\mathcal{NP} := \{\Pi : \Pi \text{ ist co-}\mathcal{NP}\text{-Problem}\}$$

$\mathcal{NP}$-Vollständigkeit

Definition ($\mathcal{NP}$-schwer, $\mathcal{NP}$-vollständig)

Entscheidungsproblem Π heißt

- $\mathcal{NP}$-schwer: jedes Problem aus $\mathcal{NP}$ polynomiell auf Π reduzierbar
- $\mathcal{NP}$-vollständig:
  - $\Pi \in \mathcal{NP}$
  - Π ist $\mathcal{NP}$-schwer

(analog für co-$\mathcal{NP}$)

Einige $\mathcal{NP}$-vollständige Probleme

- Satisfiability und 3-Sat
- Knapsack und Partition
- Stable Set und Clique
- Hamilton-Kreis und Hamilton-Pfad
- 3D-Matching